Performance Compare of JSR-94 Engines
Part 1

(contact@witmate.com)

Abstract

This is a simple performance compare between Witmate, used as JSR94 engine, and two well-known jsr94 compatible engines, JESS and Drools. JESS is a commercial package and Drools is open source. This compare is not an overall performance test. It just tries to give a basic image of Witmate performance advantage.
ATTENTION: This compare is done by Witmate, and only for our clients' system performance evaluation.

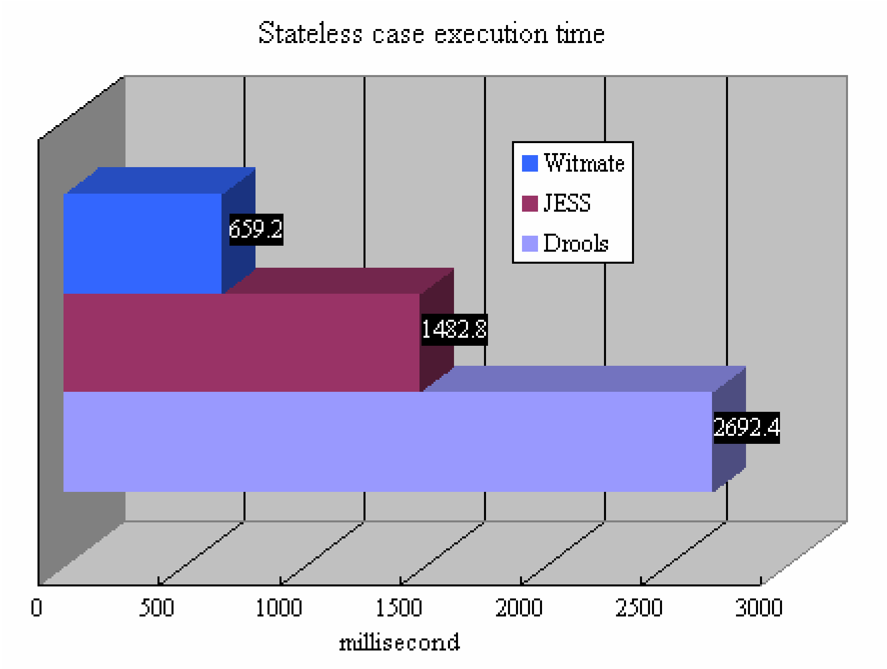
Test Case
Each JSR94 engine has their algorithm and optimization approach. To get the best performance for certain logic sets, every engine may offer very different approaches. So to get a comparable result, the test case must be general enough to cover most of usages of JSR94 API without any optimizations for special engine. Thanks JSR94 expert group, they offered a TCK to test JSR94 compatibility of engines. This TCK covered all usages of JSR94 API and without optimizations for any engines. And the TCK cases of JESS and Drools are easily found in Internet. So we chose TCK1.0 test case as this performance test case.
To get performance data, we just added system time and memory usage log code in TCK Stateless and Stateful test methods, and run each method 100 times to log total time and maximum memory usage. A batch command file boots up standalone java VM each time for each engine. And we executed this batch 10 times on test bed, then use the average value to generate compare result.

Test bed
The test bed we used is:
Hardware: P4 3GHz with HT CPU, 2G RAM
Software: Windows XP, Java 1.5.0_06
Java VM parameter: -Xnoclassgc –Xms1000m –Xmx1000m
To get correct run time and memory usage, GC (garbage collection) is closed.
Rule set data: Using XML format.
Witmate: http://witmate.com/materials/jsr94tck1.0a/tck_res_1.xml
JESS: packaged with JSR94 TCK
Drools: http://drools.org/JSR-94+TCK+Tutorial

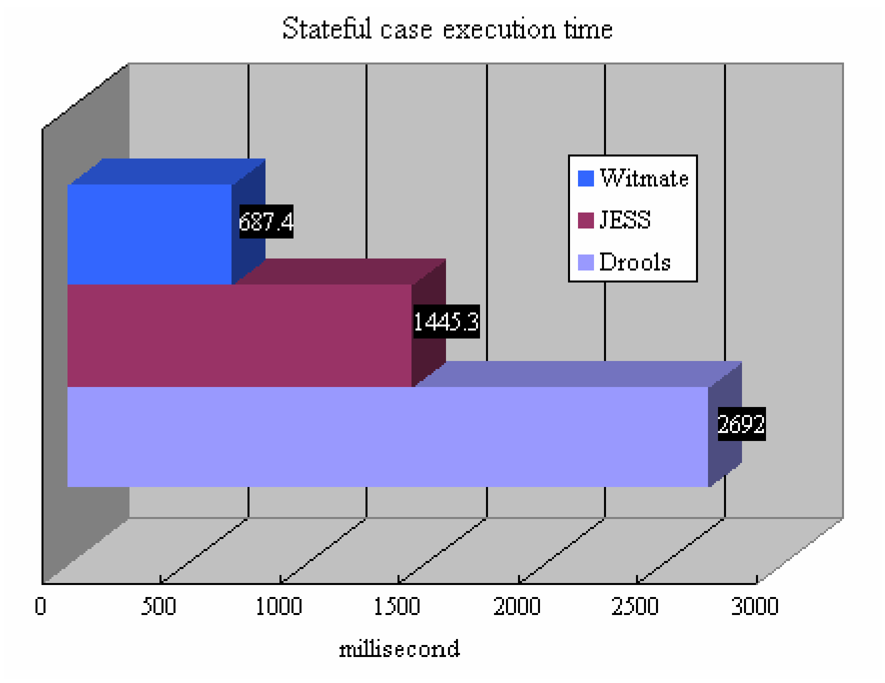
Execution time

Total execution time used by each engine. Unit is millisecond, so smaller number is faster.

Stateless case:



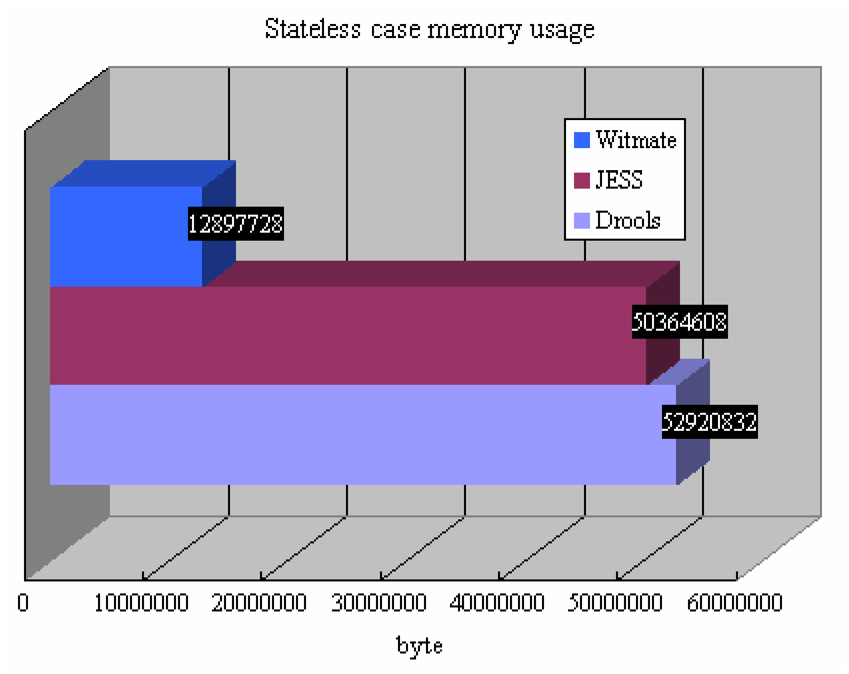
Stateful case:



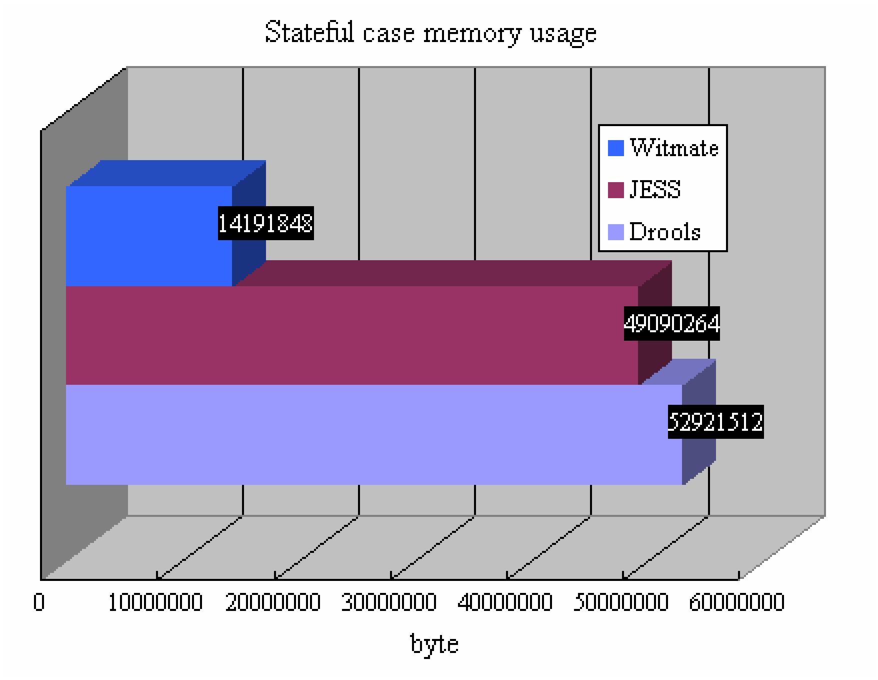
Maximum memory usage

The maximum memory used in one total execution. Smaller number figure out a smaller footprint of engine execution.

Stateless case:



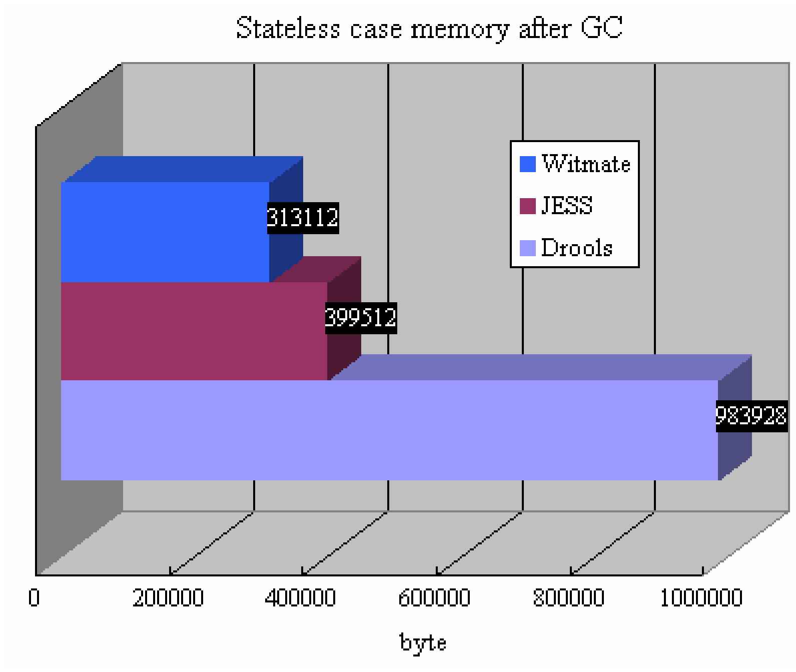
Stateful case:



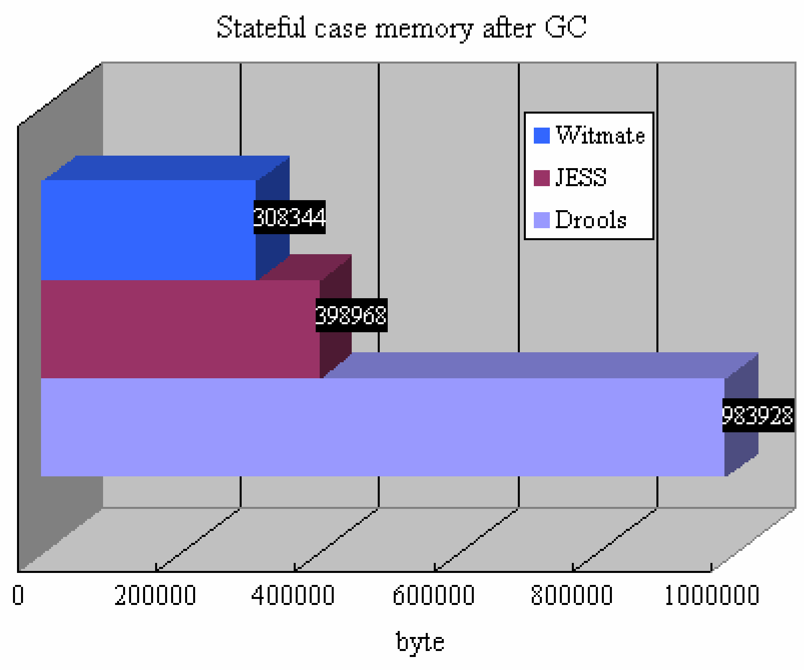
Idle status Memory usage

Idle status is the engine status that engines are not executing. To get idle status memory usage, there is a system GC calling after each execution. The engine that has smaller number of idle status memory usage uses the system resources more effectively by releasing more memory resource for executions of other processes which are running on same platform.

Stateless case:



Stateful case:



Conclusion

According to the results above, the total performance advantages of Witmate is very clear.

At runtime, Witmate performs 2-4 times of speed by 1/4 memory usage than competitors.

At idle status, the memory usage of Witmate is very small.

These advantages of Witmate give real world systems:

- 8 to 16 times of the ratio of performance/cost.
- Using rule-based application approach on mobile/small devices.
- Executing logics for any type of applications on modern complex systems.